|  |
| --- |
| , RD Dep.  MTN.\*NIX.07 Oracle DB. Introduction to DWH |
| MTN.\*NIX.07 Lab - Access and Join Methods Part 1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status of document | [**Kiryl Bucha**](mailto:Kiryl_bucha@epam.com) | 16-JAN-2012 |  |  |
|  |  |  |  |  |  |

*Contents*

[1. Table access full scan 3](#_Toc315003714)

[1.1. Task 1: Full Scans and the High-water Mark and Block reading 3](#_Toc315003715)

[2. Index Scan types 4](#_Toc315003716)

[2.1. Task 2: Index Clustering factor parameter 4](#_Toc315003717)

[2.2. Task 3: Index Unique Scan 5](#_Toc315003718)

[2.3. Task 4: Index Range Scan 5](#_Toc315003719)

[2.4. Task 5: Index Skip Scan 6](#_Toc315003720)

# Table access full scan

## Task 1: Full Scans and the High-water Mark and Block reading

Step 1:

# CREATE TABLE t2 AS

SELECT TRUNC( rownum / 100 ) id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;

Step 2:

# CREATE INDEX t2\_idx1 ON t2

( id );

**Step 3:**

Block count:

# select blocks from user\_segments where segment\_name = 'T2';

Used Block Count:

# select count(distinct (dbms\_rowid.rowid\_block\_number(rowid))) block\_ct from t2 ;

Explain Plan:

# SET autotrace ON;

# SELECT COUNT(\*) FROM t2;

Statistics

----------------------------------------------------------

0 recursive calls

0 db block gets

1541 consistent gets

0 physical reads

0 redo size

528 bytes sent via SQL\*Net to client

519 bytes received via SQL\*Net from client

2 SQL\*Net roundtrips to/from client

0 sorts (memory)

0 sorts (disk)

1. rows processed

**NOTE:** If you received next error: Check PLUSTRACE role is enabled. Please make next steps:

1. Run next script connected as sysdba:

# @ $ORACLE\_HOME/sqlplus/admin/plustrce.sql;

1. Grant role PLUSTRACE to $UserName$

# grant plustrace to buchak;

Step 4: Delete All Rows from table

# DELETE FROM t2;

**Step 5:** Repeat Step 3 and collect results.

Step 6: Insert 1 row

# INSERT INTO t2

( ID, T\_PAD )

VALUES

( 1,'1' );

COMMIT;

**Step 7:** Repeat Step 3 and collect results.

Step 8: Truncate Table

# TRUNCATE TABLE t2;

**Step 9:**  Repeat Step 3 and collect results.

**Task Results:**

Expected:

Summary table with all result and text description of analyses this results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 3 | 1664 | 1536 | 99999 | 1609 | We have 99999 rows and for scan we read 1609 times from 1536 blocks |
| 4 | 1664 | 0 | 0 | 1541 | Delete all information from table and for scan we read 1541 times.We might have many blocks that no longer contain data, but they are still under the  HWM. |
| 7 | 1664 | 1 | 1 | 1541 | One block was rewrite and for scan we read 1541 times one block.We might have many blocks that no longer contain data, but they are still under the  HWM. |
| 8 | 8 | 0 | 0 | 5 | TRUNCATE will reset the HWM of a table back to zero and will truncate the associated  indexes on the table as well. |

# Index Scan types

## Task 2: Index Clustering factor parameter

Step 1: Create table t2 as on task 1 step 1-2

Step 2: Create table t1 as listed below

# CREATE TABLE t1 AS

SELECT MOD( rownum, 100 ) id, rpad( rownum,100 ) t\_pad

FROM dual

CONNECT BY rownum < 100000;

Step 3:

# CREATE INDEX t1\_idx1 ON t1

( id );

Step 4: Calculate statistic for both tables:

# EXEC dbms\_stats.gather\_table\_stats( USER,'t1',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );

# EXEC dbms\_stats.gather\_table\_stats( USER,'t2',method\_opt=>'FOR ALL COLUMNS SIZE 1',CASCADE=>TRUE );

**Step 5:** Select Clustering Factor

# SELECT t.table\_name||'.'||i.index\_name idx\_name,

i.clustering\_factor,

t.blocks,

t.num\_rows

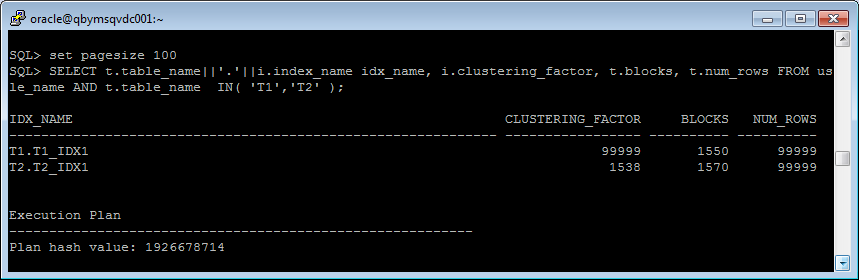
FROM user\_indexes i, user\_tables t

WHERE i.table\_name = t.table\_name

AND t.table\_name IN( 'T1','T2' );

**Task Results:**

Expected:

* Screenshot of the step 5;
* 
* Description of the parameter clustering factor;

CLUSTERING\_FACTOR is an indication of how ordered the table is with respect to the index itself how we will perform I/Os.

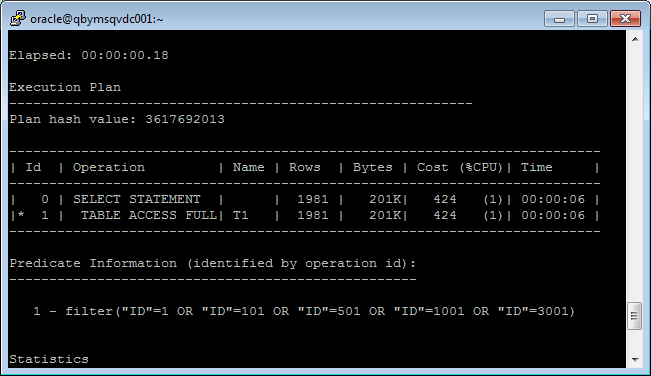
Indicates the amount of order of the rows in the table based on the values of the index:

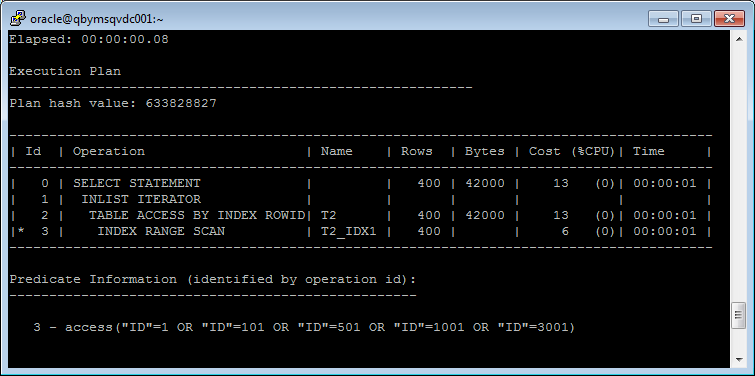
If the value is near the number of blocks, then the table is very well ordered. In this case, the index entries in a single leaf block tend to point to rows in the same data blocks.

If the value is near the number of rows, then the table is very randomly ordered. In this case, it is unlikely that index entries in the same leaf block point to rows in the same data blocks.

* Explanation: why for indexes t1\_idx1 and t2\_idx1 we have different values ;
* Which Index has best selective performance in execution Select clause filtered by IN ( , list of values, );

Index t2\_idx1 has more performance, so as him clustering factor almost equally number blocks.

* select \* from t1 where id in (1, 101, 501, 1001, 3001); 
* select \* from t2 where id in (1, 101, 501, 1001, 3001);



## Task 3: Index Unique Scan

Step 1:

# CREATE UNIQUE INDEX udx\_t1 ON t1( t\_pad );

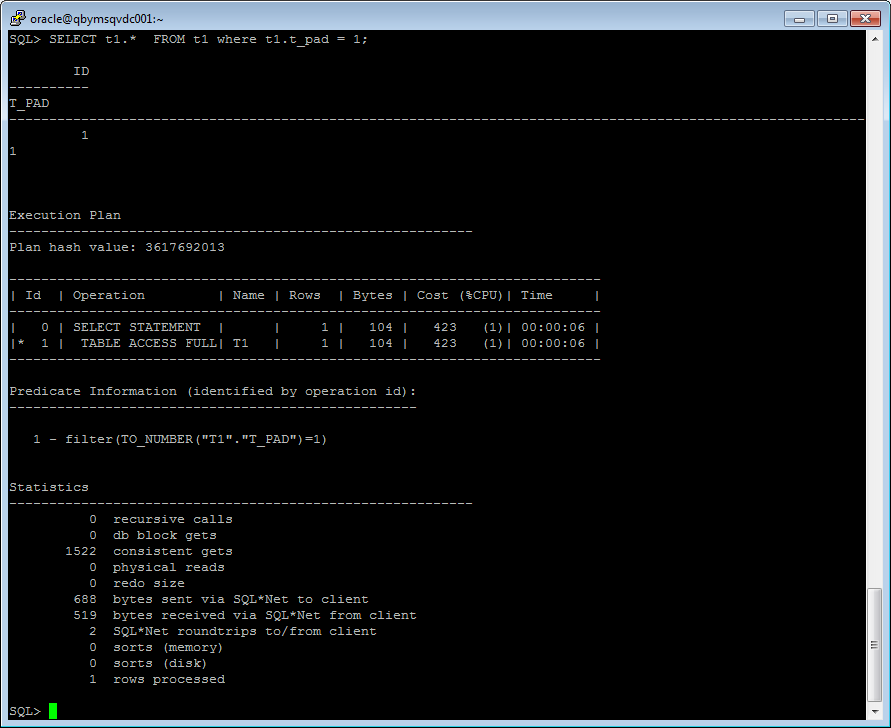
**Step 2**

# SELECT t1.\* FROM t1 where t1.t\_pad = '1';

**Task Results:**

Expected:

* Screenshot of the step 2;



* Description of process: How oracle read block on step 2;

The table access by index row id that only one row will be returned

for a specified value.

## Task 4: Index Range Scan

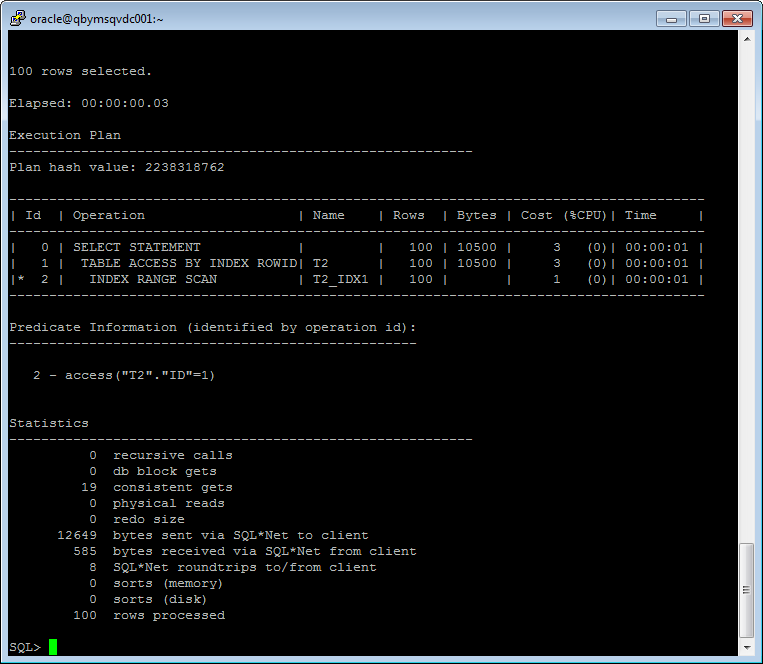
**Step 1:**

# SELECT t2.\* FROM t2 where t2.id = '1';

**Task Results:**

Expected:

* Screenshot of the step 1;



* Description of process: How oracle read block on step 1;

An index range scan is chosen when a predicate contains a condition that will return a range of data.

The index can be unique or non-unique as it is the condition that determines whether or not multiple

rows will be returned or not. The conditions specified can use operators such as <, >, LIKE, BETWEEN and even =.

## Task 5: Index Skip Scan

Step 1:

# CREATE TABLE employees AS

SELECT \*

FROM scott.emp;

Step 2:

# CREATE INDEX idx\_emp01 ON employees

( empno, ename, job );

**Step 3:**  Get trace and statistic of explain plan

# SELECT /\*+INDEX\_SS(emp idx\_emp01)\*/ emp.\* FROM employees emp where ename = 'SCOTT';

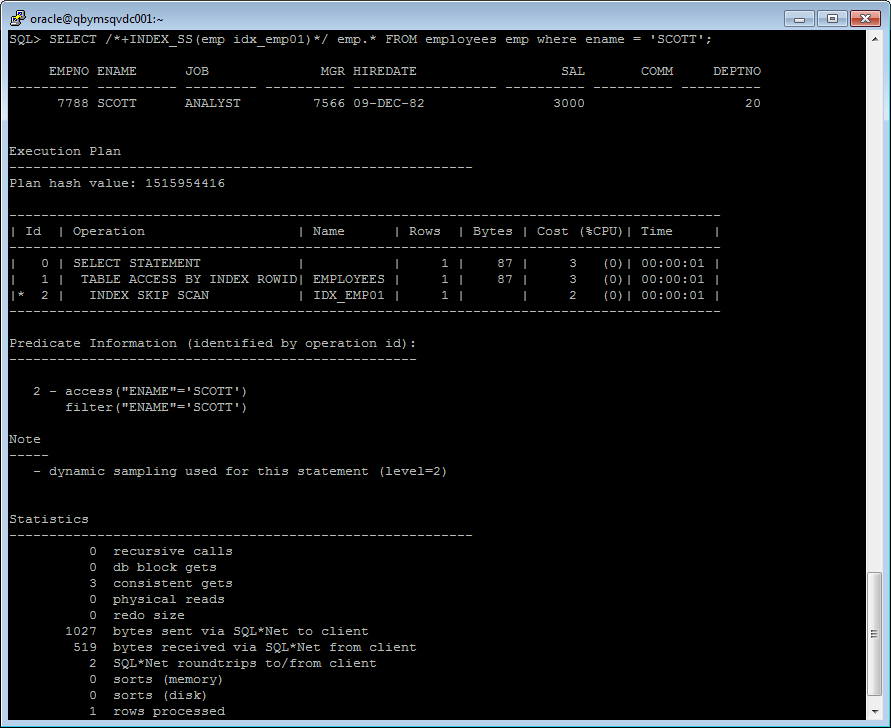
# SELECT /\*+FULL\*/ emp.\* FROM employees emp WHERE ename = 'SCOTT';

**Task Results:**

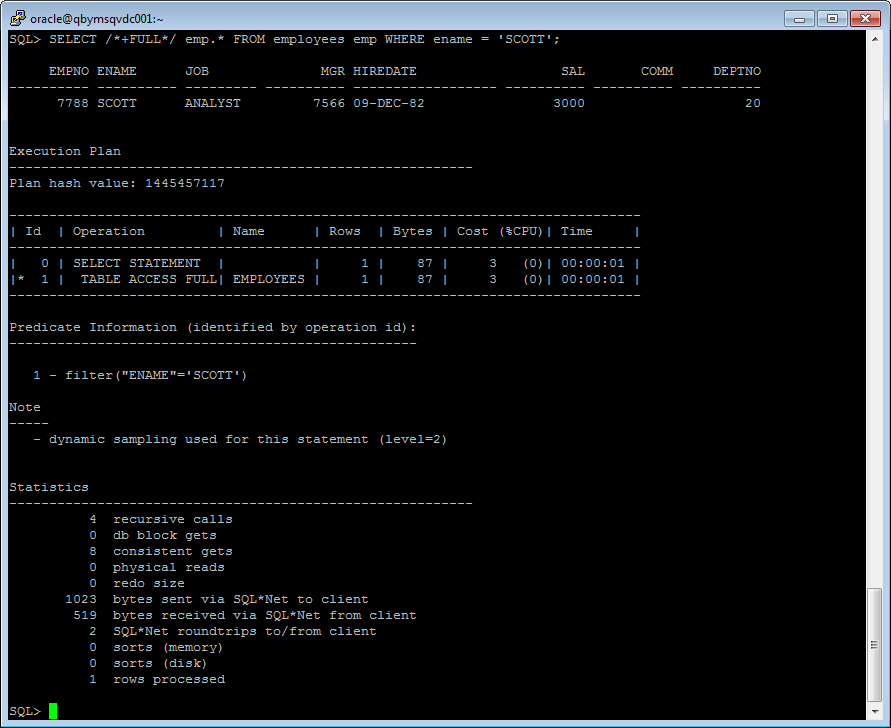
Expected:

* 2 Screenshots of the step 3;

SELECT /\*+INDEX\_SS(emp idx\_emp01)\*/ emp.\* FROM employees emp where ename = 'SCOTT';



SELECT /\*+FULL\*/ emp.\* FROM employees emp WHERE ename = 'SCOTT';



* Description of process: How oracle analyses index that was created on step 2;

A **skip scan** works by logically splitting a multi-column index into smaller subindexes. The number of logical subindexes is determined by the number of distinct values in the leading columns of the index. Therefore, the more distinct the leading columns are, the more logical subindexes would need to be created. If too many subindexes would be required, the operation won’t be as efficient as simply doing a full scan.

However, in the cases where the number of subindexes needed would be smaller, the operation can be

many times more efficient than a full scan as scanning smaller index blocks can be more efficient

than scanning larger table blocks.

An **index full scan** is chosen under several conditions including: when there is no predicate but the

column list can be satisfied through an index on a column, the predicate contains a condition on a

non-leading column in an index, or the data can be retrieved via an index in sorted order and save the

need for a separate sort step.

* Summary table with all result and text description of analyses this results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Count of Blocks | Count of Used Blocks | Count of Rows | Consistent gets | Description |
| 3 | 1664 | 1 | 99999 | 1522 | Index unique scan |
| 4 | 1664 | 3 | 99999 | 19 | Index range scan |
| 5 | 8 | 1 | 14 | 3 | skip scan |
| 5 | 8 | 4 | 14 | 8 | index full scan |